




Southeast University



软件体系结构概述

王璐璐 wanglulu@seu.edu.cn

廖力 lliao@seu.edu.cn



第1章: 概 论

- (一) 软件架构产生的背景
- (二) 软件架构的主要思想和特征
- (三) 软件架构的发展阶段
- (四) 软件架构的研究和应用现状

一、软件架构产生的背景

- 1. 软件危机

- **20世纪60-90年代** 软件生产效率低下, 软件产品质量差。软件开发理论不够系统、技术手段相对滞后, 手工作坊式生产。

Architecting a dog house

Can be developed or build by one person as it requires:

- Minimal modeling
- Simple Process
- Simple tools



一、软件架构产生的背景

- 1. 软件危机

- **20世纪90年代后** 随软件规模的进一步扩大和软件复杂性的不断提高，新一轮软件危机再次出现。

Architecting a house



Built most efficiently and timely by a team. It requires:-

- Modeling
- Well-defined processes
- Power tools

What About Skyscrapers ?



Differences

- Scale
- Process
- Cost
- Schedule
- Skills and development teams
- Material and technologies
- Stakeholders
- Risks

一、软件架构产生的背景

- 1. 软件危机

- 危机根源

- (1) 软件复杂、易变，其行为特性难以预见，软件开发过程中需求向设计缺乏有效的转换，导致软件开发过程的困难和不可控；
- (2) 随着软件系统规模越来越大、越来越复杂，整个系统的结构和规格说明显得越来越重要；
- (3) 对于大规模的复杂软件系统，总体的系统结构设计和规格说明比起对计算的算法和数据结构的选择已经变得明显重要得多；
- (4) 对软件系统结构的深入研究将会成为提高软件生产率和解决软件维护问题的最有希望的新途径。

一、软件架构产生的背景

- 2.软件架构的产生

- **20世纪90年代**， 研究人员展开了关于软件架构的基础研究，主要集中在架构风格(模式)、架构描述语言、架构文档和形式化方法。
- 软件架构在高级层次上对软件进行描述，便于软件开发过程中各个视角（例如用户、业务和系统）的统一，能够及早发现开发中的问题并支持各种解决方案的评估和预测。
- 其意义贯穿软件生命周期的各个阶段。

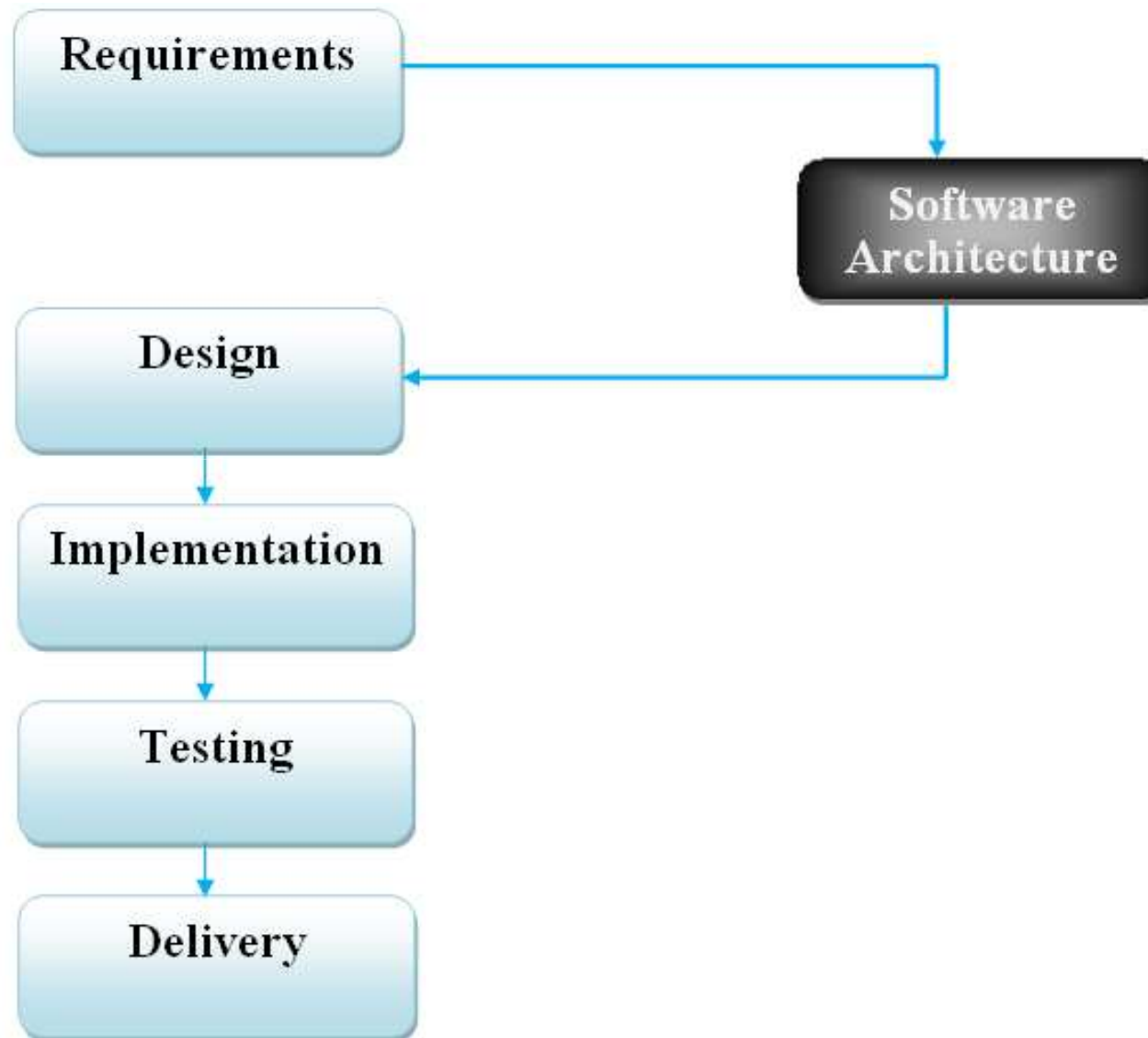
The Need of Architecture

The Winchester “Mystery” House

- 38 years of construction – 147 builders 0 architects
- 160 rooms – 40 bedrooms, 6 kitchens, 2 basements, 950 doors
- 65 doors to blank walls, 13 staircases abandoned, 24 skylights in floors
- No architectural blueprint exists



Software Architecture in Development





二、软件架构的主要思想和特征

- 1. 主要思想

- 软件架构是一个软件系统的设计图，并不仅限于软件系统的总体结构，还包含一些质量属性以及功能与结构之间的映射关系，即设计决策。
- 软件架构的两个主要焦点集中于系统的总体结构以及需求和实现之间的对应。

二、软件架构的主要思想和特征

- 1. 主要思想

- 软件架构的主要思想是将注意力集中在系统总体结构的组织上。
- 实现的手段是运用抽象方法屏蔽错综复杂的模块间连接，使人们的认知提升并保持在整体结构的部件“交互”层次，并进一步将交互从计算中分离出来，建立“组件+连接件+配置”的软件系统高层结构组织方式。

二、软件架构的主要思想和特征

- 2.软件架构的特征

- (1) 注重可重用性——组件及架构级重用
- (2) 利益相关者较多——平衡需求
- (3) 关注点分离——模块化、分而治之
- (4) 质量驱动——关注非功能属性
- (5) 提倡概念完整性——强调设计决策是一个持续的过程
- (6) 循环风格——用标准方法来处理反复出现的问题

三、软件架构的发展阶段

- 基础研究阶段（1968-1994）
- 概念体系和核心技术形成阶段（1991-2000）
- 理论体系丰富发展阶段（1996-1999）
- 理论完善和普及应用阶段（1999年至今）

三、软件架构的发展阶段

- 1. 基础研究阶段（1968-1994）
 - 1968北大西洋公约组织(NATO)会议上第一次出现词语“软件架构”
 - 直到20世纪80年代，“架构”一词大部分情况下被用于表示计算机系统的物理结构，偶尔被用于表示计算指令集的特定体系。
 - 随着软件规模增大，开发者已经开始尝试模块化（modularization）的实践

三、软件架构的发展阶段

- 1. 基础研究阶段 (1968-1994)
 - 模块化:指的是一种软件开发方法, 把一个待开发的软件分解成若干小的简单的部分, 称为模块。每一个模块都独立地开发、测试, 最后再组装出整个软件。
 - 模块化规则:
 - 高内聚, 低耦合
 - 模块大小适度
 - 模块调用链的深度 (嵌套层次)不可过多
 - 接口干净, 信息隐蔽
 - 尽可能地复用已有模块

三、软件架构的发展阶段

- 2.概念体系和核心技术形成阶段（1991-2000）
 - 1991年，Winston W. Royce 与 Walker Royce 首次对软件架构进行了定义。
 - 1992年D E.Perry与A L.Wolf对软件架构进行了阐述，提出了著名的{elements, forms, rationale} = software architecture公式
 - 1996年CMU/SEI的Mary Shaw和David Garlan，出版了《Software Architecture: Perspectives on an Emerging Discipline》，对软件架构概念的内涵与外延进行了详尽阐述，对软件架构概念的形成起到了至关重要的作用。

三、软件架构的发展阶段

- 2.概念体系和核心技术形成阶段（1991-2000）
 - 从1995年起，软件架构研究领域开始进入快速发展阶段
 - Booch, Rumbaugh and Jacobson从另一个角度对软件架构的概念进行了全新的诠释，认为架构是一系列重要决策的集合。
 - 提出了软件架构实践方法体系SAAM；
 - Rechtin 与 Mark Maier在1998年出版的《The Art of Systems Architecting》很好地阐述了系统与软件的关系。

三、软件架构的发展阶段

- 2.概念体系和核心技术形成阶段（1991-2000）
 - 这一阶段最重要的成果之一就是软件组件（component）技术
 - 组件化开发并不等同于模块化开发。模块化开发只是在逻辑上做了切分，物理上（开发出的系统代码）通常并没有真正意义上的隔离。
 - 组件化也不等同于应用集成，应用集成是将一些基于不同平台或不同方案的应用软件和系统有机地集成到一个无缝的、并列的、易于访问的单一系统中，建立一个统一的综合应用。
 - 组件化应比模块化更独立，但比应用集成结合得更加紧密。

三、软件架构的发展阶段

- 3.理论体系丰富发展阶段（1996-1999）
 - 随着基于组件软件架构理论的建立，与之相关的一些研究方向逐渐成为软件工程领域的研究热点。主要研究方向包括：
 - 软件架构的描述与表示；
 - 软件架构分析、设计与测试；
 - 软件架构发现、演化与重用；
 - 基于软件架构的开发方法；
 - 软件架构的风格
 - 等等

三、软件架构的发展阶段

- 4.理论完善和普及应用阶段（1999-至今）
 - 1999年，第一届IFIP软件架构会议召开；Open Group 提出了架构描述Markup语言，一种基于XML的架构描述语言，支持广泛的架构模型共享；软件产品线成为了软件架构的一个重要分支，吸引了大量的大型企业的关注。
 - 2000年，IEEE 1471-2000发布，为软件架构的普及应用制定了标准化规范，该标准随后分别于2007年与2011年得到扩充与修改。
 - 2003年，《Software Architecture in Practice》一书出版。

四、软件架构研究和应用现状

- 1. 软件架构理论和方法研究

- (1) 软件架构描述与构造表示

- 形式化的架构描述方法：各种ADL的出现
 - Kruchten的“4+1”架构模型：从5个不同的视角包括逻辑视角、过程视角、物理视角、开发视角和场景视角来描述软件架构。
 - 使用UML的架构描述方法
 - IEEE的软件架构描述规范（IEEE 1471-2000）

四、软件架构研究和应用现状

- 1.软件架构理论和方法研究

- (2) 软件架构分析、设计与测试

- 架构分析的内容可分为结构分析、功能分析和非功能分析。
 - 软件架构分析的目的是在系统被实际构造之前，预测其质量属性。
 - 软件架构分析常见方法如：基于场景的架构分析方法SAAM、架构折中分析方法ATAM等

四、软件架构研究和应用现状

- 1. 软件架构理论和方法研究

- (2) 软件架构分析、设计与测试

- 软件架构的设计指生成一个满足软件需求的架构的过程。
 - 常见的软件架构的设计方法如
 - 工件驱动 (artifact-driven) 方法
 - 用例驱动 (usecase-driven) 方法
 - 模式驱动 (pattern-driven) 方法
 - 领域驱动 (domain-driven) 方法
 - 属性驱动设计 (Attribute-Driven Design) 方法等

四、软件架构研究和应用现状

- 1. 软件架构理论和方法研究

- (2) 软件架构分析、设计与测试

- 架构测试着重于仿真系统模型，解决架构层的主要问题。
 - 由于测试的抽象层次不同,架构测试策略可以分为单元/子系统/集成/验收测试等阶段的测试策略。
 - 架构测试技术主要包括：
 - Debra等人提出一组针对架构的测试覆盖准则，如组件覆盖准则等
 - 基于霍尔公理的组件设计正确性验证技术
 - 基于CHAM（CHemical Abstract Machine）的架构动态语义验证技术等

四、软件架构研究和应用现状

- 1. 软件架构理论和方法研究

- (3) 软件架构发现、演化与复用

- 架构发现解决如何从已经存在的系统中提取软件的架构，属于逆向工程。
 - 软件架构的演化值由于系统需求、技术、环境、分布等因素的变化而最终导致软件架构的变动。
 - 架构重用属于设计重用，比代码重用更抽象。架构模式就是架构复用的一个研究成果。

四、软件架构研究和应用现状

- 1. 软件架构理论和方法研究

- (4) 基于软件架构的开发模型

- 跨越整个软件生存周期的系统开发、运行、维护所实施的全部工作和任务的结构框架

- (5) 软件架构的风格与模式

- 架构风格（架构模式）是对给定场景中经常出现的问题提供的一般性的可重用方案

四、软件架构研究和应用现状

- 1. 软件架构理论和方法研究

- (5) 软件架构的风格与模式

- David Garlan和Mary Shaw等人将被广泛接受的架构风格分成五种主要的类型：
 - 1) 数据流风格：批处理序列；管道-过滤器
 - 2) 调用/返回风格：主程序/子程序；面向对象风格；层次结构
 - 3) 独立组件风格：进程通信；事件系统
 - 4) 虚拟机风格：解释器；基于规则的系统
 - 5) 仓库风格：数据库系统；超文本系统；黑板系统

四、软件架构研究和应用现状

- 1. 软件架构理论和方法研究

- (6) 软件产品线架构

- 软件产品线表示着一组具有公共的系统需求集的软件系统，它们都是根据基本的用户需求对标准的产品线架构进行定制，将可重用组件与系统独有的部分集成而得到的。
 - 软件产品线架构的发展是依托着特定领域软件架构（Domain Specific Software Architecture, DSSA）的研究深入而进行的。
 - 如：北京大学杨芙清院士牵头实现的“支持组件复用的青鸟III型系统”等。

四、软件架构研究和应用现状

- 1.软件架构理论和方法研究
 - (7) 软件架构支持工具
 - 支持静态分析的工具
 - 支持类型检查的工具
 - 支持架构层次依赖分析的工具
 - 支持架构动态特性仿真的工具
 - 支持架构性能仿真的工具等

四、软件架构研究和应用现状

- 2.软件架构的应用研究

- (1) 软件架构风格的应用

- 不同的架构风格具有各自的优缺点和应用场景
 - 例如：
 - 虚拟机风格经常用于构造解释器或专家系统；
 - C/S和B/S适合于数据和处理分布在一定范围、通过网络连接构成的系统；
 - “平台-插件”风格适用于具有插件扩展功能的应用程序中；
 - MVC被广泛的应用于用户交互程序的设计；
 - SOA应用在企业集成等方面

四、软件架构研究和应用现状

- 2.软件架构的应用研究
 - (2) 软件架构在开发过程中的应用
 - 需求阶段：把SA的概念引入到需求分析阶段，有助于保证需求规约和系统设计之间的可追踪性和一致性。
 - 设计阶段：SA模型的描述、SA模型的设计与分析方法，以及对SA设计经验的总结与复用等。
 - 实现阶段：将设计阶段设计的算法及数据类型用程序设计语言进行表示
 - 维护阶段：为了保证软件具有良好的维护性，在软件架构中针对维护性的目标进行分析时，需要对一些有关维护性的属性（例如，可扩展性、可替换性等）进行规定。